REMARKS

Claims 1 – 15 are pending in the present application.

Claim 14 stood rejected under 35 U.S.C. §112, 2nd paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the Applicant regards as the invention. Claim 14 has been amended to overcome the Examiner's rejection.

Claims 1 – 15 stand rejected under 35 U.S.C. §102(b) as being anticipated by May et al. (US Patent Number 4,783,734) (hereinafter 'May'). Applicant respectfully traverses these rejections.

Claims 3-4, 8-9, and 13-14 stand rejected under 35 U.S.C. §103(a) as being unpatentable over May in view of Conger "Windows API Bible 1992" (hereinafter 'Conger'). Applicant respectfully traverses these rejections.

The applicant discloses at page 5, line 23 through page 6, line 26

> "If the process 12(n')(m') is waiting for information from the process 12(n)(m), which may occur after it issues a processing request to the process 12(n)(m), it (that is, process 12(n')(m')) will perform a "spin-wait" loop in which it periodically tests the flag 16(n')(m') while performing the spin-wait loop, the process 12(n')(m') will need to delay performing other processing operations until the process 12(n)(m) has loaded the information in its memory portion 13(n')(m')and set the flag 16(n')(m'). ... To reduce the amount of processing time that may be wasted by the process 12(n')(m') executing the spin loop, the computer 10 also includes a spin daemon 17 which, after receiving a flag monitor request therefor from the process 12(n')(m'), can monitor the condition of the flag 16(n')(m') for the process 12(n')(m'). The process 12(n')(m'), after issuing a flag monitor request to the spin daemon 17, can thereafter de-schedule itself, by removing its identification from the task list 15, and provide a notification to the operating-system 14 giving up any remaining portion of its current time slot. ... After receiving the flag monitor request from the process 12(n')(m'), the spin daemon 17 will add the flag monitor request to a spin list 18 which it maintains. The spin list 18 identifies the flags in memory whose condition it is to monitor. Accordingly, after the

spin daemon 17 adds the flag monitor request to the spin list 18, it will thereafter monitor the condition of the flag 16(n')(m'). When the process 12(n)(m) sets the flag, indicating that has provided the information required by the process 12(n')(m'), the spin daemon 17 will enable the process 12(n')(m') to, in turn, enable the operating system 14 to re-schedule the process 12(n')(m') by reloading its identification on the operating-system task list 16."

Accordingly, the Applicant's claim 1 recites

"a system for controlling co-scheduling of processes in a computer comprising at least one process and a spin daemon, the process being configured to, when it is waiting for a flag to change condition, transmit a flag monitor request to the spin daemon and de-schedule itself, the spin daemon being configured to, after receiving a flag monitor request monitor the flag and, after the flag changes condition, enable the at least one process to be re-scheduled for execution by the computer."

Applicants claim 2 recites

"said spin daemon is configured to monitor a plurality of flags, each in response to a flag monitor request, the spin daemon maintaining a list identifying those flags it is to monitor, the spin daemon being further configured to, when it receives a flag monitor request, add an identification of a flag associated with the request to the list."

The Examiner asserts that May teaches a process and a spin daemon (col. 39, lines 24-41), wherein the process is waiting for a flag (PRIFLAG 47 col. 34, line 45) to change condition (test condition col. 34, lines 55-67 and PRO Run col. 19 lines 47-52), wherein the process transmits a flag monitor request (added to list or queue col. 31, lines 52-56) to the spin daemon and de-schedule itself (col. 31 lines 41-62), and after the flag changes condition (col. 34 lines 41-68), enable the process to be rescheduled (col. 34 lines 67-68) for execution by the computer. Applicant respectfully disagrees with the Examiner's characterization of May.

May is directed toward a method and system for communicating variable length data between a plurality of concurrent processes. Specifically, at col. 39, lines 25 – 39 May discloses

"An alternative process is one which selects one of a number of channels for input and then executes a corresponding component process of the

alternative process. The selection of the channel from the alternatives available is performed by **the inputting process examining all the channels to determine if one or more are ready for input in the sense that it has an outputting process already waiting to output through that channel**. If no channel is found to be ready the inputting process is descheduled until one of the channels becomes ready. If at least one of the channels was found to be ready, or if the inputting process is rescheduled due to action by an outputting process, the process selects one of the inputs which is now ready and performs the input through that channel."

Thus, it appears that the inputting process of May examines several input channels for a ready channel and de-schedules itself if there are no ready channels. The outputting process appears to be just an output process which may use the channel selected by the inputting process. Further, it appears that the inputting channel may become re-scheduled "due to action by an outputting process" and not by a separate monitoring process. Thus, the outputting process of May **is not analogous** to a spin daemon.

In addition, May discloses at col. 31, lines 52-56 "Any process which is not the current process and is not awaiting execution is descheduled. When a process is scheduled it either becomes the current process or is added to a list or queue of processes awaiting execution." Applicant submits that **this is not analogous** to a process transmitting a flag monitor request to a spin daemon as suggested by the examiner.

Further, May discloses at col. 21, lines 34-48

"The processor performs a sequence of actions. These are performed either on behalf of the current process or on behalf of a link.
The actions which may be performed on behalf of the current process are to perform "StartNextProcess", to perform "BlockCopyStep" or to fetch, decode and execute an instruction.
...The last microinstruction in any of the sequences comprising these actions is "NextAction". This causes the processor to choose the next action to be performed.
The way in which the processor decides which action is to be performed next when a "NextAction" microinstruction is executed is as described below.
The Sync Control Logic 10 will forward at most one "RunRequest" or "ReadyRequest" to the processor at any time. The Sync Control Logic will not forward a priority 1 request if there is a priority 0 request outstanding.

> This results in two signals entering the Condition Multiplexor, one indicating the presence of a request, the other indicating the priority of that request.
> The Condition Multiplexor also has signals coming from the currently selected SNPFlag and the currently selected CopyFlag. It is, therefore, able to make the selection as described below.
> The processor will perform "StartNextProcess" if the SNPFlag[Pri] is set. Otherwise, the processor will handle a channel request, unless the priority of that request is lower than the priority of the current process. Otherwise, the processor will perform "BlockCopyStep" if the CopyFlag[Pri] is set. Otherwise the processor will fetch, decode and execute an instruction if there is a current process. Otherwise the processor will wait until there is a channel request."

From the foregoing, it appears that May uses a series of procedure calls executed by the processor to schedule and deschedule processes in response to control signals generated by sync control logic 10. Applicant submits that this is in contrast to using a separate process (i.e. spin daemon) to monitor flags that are in a list and which enables de-scheduled processes to be re-scheduled.

Accordingly, May does not teach or disclose "at least one process and a spin daemon, the process being configured to, when it is waiting for a flag to change condition, transmit a flag monitor request to the spin daemon and de-schedule itself, the spin daemon being configured to, after receiving a flag monitor request monitor the flag and, after the flag changes condition, enable the at least one process to be re-scheduled for execution by the computer" as recited in Applicant's claim 1.

The Applicant submits that claim 1, along with its dependent claims 2 – 5, are believed to patentably distinguish over May and over May in view of Conger for the reasons given above.

Claims 6 and 11 recite features similar to claim 1. As such, claims 6 and 11, along with their respective dependent claims 7 – 10 and 12 – 15, are also believed to patentably distinguish over May and over May in view of Conger for at least the same reasons.

CONCLUSION

Applicant submitS the application is in condition for allowance, and an early notice to that effect is requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-93900/BNK.

Respectfully submitted,

Stephen J. Curran
Reg. No. 50,664
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: 2-4-2004